PART I CONTEMPORARY SECURITY

1 The Need for Secure Systems

Applications on the Wild Wild Web

The Need for Trustworthy Computing

Getting Everyone's Head in the Game

Using Tact to Sell Security to the Organization

Using Subversion

Some Ideas for Instilling a Security Culture

Get the Boss to Send an E-Mail

Nominate a Security Evangelist

The Attacker's Advantage and the Defender's Dilemma

Principle #1: The defender must defend all points; the attacker can choose the weakest point.

Principle #2: The defender can defend only against known attacks; the attacker can probe for unknown vulnerabilities.

Principle #3: The defender must be constantly vigilant; the attacker can strike at will.

Principle #4: The defender must play by the rules; the attacker can play dirty.

Summary

2 The Proactive Security Development Process

Process Improvements

The Role of Education

Resistance to Mandatory Training

Ongoing Training

Advancing the Science of Security

Education Proves the More Eyes Fallacy

Now the Evidence!

Design Phase

Security Questions During Interviews

Define the Product Security Goals

Security Is a Product Feature

Making Time for Security

Threat Modeling Leads to Secure Design

Build End-of-Life Plans for Insecure Features

Setting the Bug Bar

Security Team Review

Development Phase

Be Hardcore About Who Can Check In New Code (Check-Ins Checked)

Security Peer Review of New Code (Check-Ins Checked)

Define Secure Coding Guidelines

Review Old Defects

BUY ONLINE AT: http://www.itgovernance.co.uk/products/747
### PART I: AUTHENTICATION, AUTHORIZATION, AND ACCESS CONTROL

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>109</td>
</tr>
<tr>
<td>Authorization</td>
<td>114</td>
</tr>
<tr>
<td>Tamper-Resistant and Privacy-Enhanced Technologies</td>
<td>115</td>
</tr>
<tr>
<td>Protect Secrets, or Better Yet, Don't Store Secrets</td>
<td>116</td>
</tr>
<tr>
<td>Encryption, Hashes, MACs, and Digital Signatures</td>
<td>116</td>
</tr>
<tr>
<td>Auditing</td>
<td>117</td>
</tr>
<tr>
<td>Filtering, Throttling, and Quality of Service</td>
<td>118</td>
</tr>
<tr>
<td>Least Privilege</td>
<td>118</td>
</tr>
<tr>
<td>Mitigating the Sample Payroll Application Threats</td>
<td>118</td>
</tr>
<tr>
<td>A Cornucopia of Threats and Solutions</td>
<td>120</td>
</tr>
<tr>
<td>Summary</td>
<td>124</td>
</tr>
</tbody>
</table>

### PART II: SECURE CODING TECHNIQUES

#### 5 Public Enemy #1: The Buffer Overrun

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stack Overruns</td>
<td>127</td>
</tr>
<tr>
<td>Heap Overruns</td>
<td>129</td>
</tr>
<tr>
<td>Array Indexing Errors</td>
<td>138</td>
</tr>
<tr>
<td>Format String Bugs</td>
<td>144</td>
</tr>
<tr>
<td>Unicode and ANSI Buffer Size Mismatches</td>
<td>147</td>
</tr>
<tr>
<td>A Real Unicode Bug Example</td>
<td>153</td>
</tr>
<tr>
<td>Preventing Buffer Overruns</td>
<td>155</td>
</tr>
<tr>
<td>Safe String Handling</td>
<td>156</td>
</tr>
<tr>
<td>A Word of Caution About String-Handling Functions</td>
<td>159</td>
</tr>
<tr>
<td>The Visual C++ .NET /GS Option</td>
<td>167</td>
</tr>
<tr>
<td>Summary</td>
<td>170</td>
</tr>
</tbody>
</table>

#### 6 Determining Appropriate Access Control

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Why ACLs Are Important</td>
<td>171</td>
</tr>
<tr>
<td>A Diversion: Fixing the Registry Code</td>
<td>171</td>
</tr>
<tr>
<td>What Makes Up an ACL?</td>
<td>173</td>
</tr>
<tr>
<td>A Method of Choosing Good ACLs</td>
<td>178</td>
</tr>
<tr>
<td>Effective Deny ACEs</td>
<td>180</td>
</tr>
<tr>
<td>Creating ACLs</td>
<td>181</td>
</tr>
<tr>
<td>Creating ACLs in Windows NT 4</td>
<td>181</td>
</tr>
<tr>
<td>Creating ACLs in Windows 2000</td>
<td>185</td>
</tr>
<tr>
<td>Creating ACLs with Active Template Library</td>
<td>189</td>
</tr>
<tr>
<td>Getting the ACE Order Right</td>
<td>191</td>
</tr>
<tr>
<td>Be Wary of the Terminal Server and Remote Desktop SIDs</td>
<td>193</td>
</tr>
<tr>
<td>NULL DACLs and Other Dangerous ACE Types</td>
<td>195</td>
</tr>
<tr>
<td>NULL DACLs and Auditing</td>
<td>197</td>
</tr>
<tr>
<td>Dangerous ACE Types</td>
<td>197</td>
</tr>
<tr>
<td>What If I Can't Change the NULL DACL?</td>
<td>198</td>
</tr>
<tr>
<td>Other Access Control Mechanisms</td>
<td>199</td>
</tr>
</tbody>
</table>

BUY ONLINE AT: [http://www.itgovernance.co.uk/products/747](http://www.itgovernance.co.uk/products/747)
Canonical Web-Based Issues 373
Bypassing AOL Parental Controls 373
Bypassing eEye's Security Checks 374
Zones and the Internet Explorer 4 "Dotless-IP Address" Bug 374
Internet Information Server 4.0 ::$DATA Vulnerability 375
When is a Line Really Two Lines? 377
Yet Another Web Issue—Escaping 378
Visual Equivalence Attacks and the Homograph Attack 382
Preventing Canonicalization Mistakes 383
Don't Make Decisions Based on Names 383
Use a Regular Expression to Restrict What's Allowed in a Name 383
Stopping 8.3 Filename Generation 385
Don't Trust the PATH—Use Full Path Names 385
Attempt to Canonicalize the Name 386
Calling CreateFile Safely 390
Web-Based Canonicalization Remedies 391
Restrict What Is Valid Input 391
Be Careful When Dealing with UTF-8 391
ISAPIs—Between a Rock and a Hard Place 392
A Final Thought: Non-File-Based Canonicalization Issues 393
Server Names 393
Usernames 394
Summary 396

12 Database Input Issues 397
The Issue 398
Pseudoremedy #1: Quoting the Input 401
Pseudoremedy #2: Use Stored Procedures 402
Remedy #1: Never Ever Connect as sysadmin 403
Remedy #2: Building SQL Statements Securely 404
Building SQL Stored Procedures Securely 406
An In-Depth Defense in Depth Example 407
Summary 411

13 Web-Specific Input Issues 413
Cross-Site Scripting: When Output Turns Bad 413
Sometimes the Attacker Doesn't Need a <SCRIPT> Block 417
The Attacker Doesn't Need the User to Click a Link! 418
Other XSS-Related Attacks 418
XSS Attacks Against Local Files 418
XSS Attacks Against HTML Resources 420
XSS Remedies 421
Encoding Output 422

BUY ONLINE AT: http://www.itgovernance.co.uk/products/747
Adding Double Quotes Around All Tag Properties 422
Inserting Data in the innerText Property 423
Forcing the Codepage 423
The Internet Explorer 6.0 SP1 HttpOnly Cookie Option 424
Internet Explorer "Mark of the Web" 425
Internet Explorer <FRAME SECURITY> Attribute 426
ASP.NET 1.1 ValidateRequest configuration option 427
Don't Look for Insecure Constructs 428

But I Want Users to Post HTML to My Web Site! 430
How to Review Code for XSS Bugs 431
Other Web-Based Security Topics 431
eval() Can Be Bad 431
HTTP Trust Issues 432
ISAPI Applications and Filters 433
Be Wary of "Predictable Cookies" 436
SSL/TLS Client Issues 437
Summary 438

14 Internationalization Issues 439
The Golden I18N Security Rules 440
Use Unicode in Your Application 440
Prevent I18N Buffer Overruns 441
Words and Bytes 442
Validate I18N 443
Visual Validation 443
Do Not Validate Strings with LCMapString 443
Use CreateFile to Validate Filenames 443
Character Set Conversion Issues 444
Use MultiByteToWideChar with MB_PRECOMPOSED and MB_ERR_INVALID_CHARS 444
Use WideCharToMultiByte with WC_NO_BEST_FIT_CHARS 445
Comparison and Sorting 448
Unicode Character Properties 448
Normalization 450
Summary 451

PART III EVEN MORE SECURE CODING TECHNIQUES 455
15 Socket Security 455
Avoiding Server Hijacking 456
TCP Window Attacks 463
Choosing Server Interfaces 464
Accepting Connections 464
Writing Firewall-Friendly Applications 470

BUY ONLINE AT: http://www.itgovernance.co.uk/products/747
Use One Connection to Do the Job 471
Don't Require the Server to Connect Back to the Client 471
Use Connection-Based Protocols 472
Don't Multiplex Your Application over Another Protocol 472
Don't Embed Host IP Addresses in Application-Layer Data 473
Make Your Application Configurable 473
Spoofing and Host-Based and Port-Based Trust 473
IPv6 Is Coming! 474
Summary 476

16 Securing RPC, ActiveX Controls, and DCOM 477
An RPC Primer 478
What Is RPC? 478
Creating RPC Applications 479
How RPC Applications Communicate 481
Secure RPC Best Practices 482
Use the /robust MIDL Switch 483
Use the [range] Attribute 483
Require Authenticated Connections 484
Use Packet Privacy and Integrity 489
Use Strict Context Handles 491
Don't Rely on Context Handles for Access Checks 492
Be Wary of NULL Context Handles 493
Don't Trust Your Peer 494
Use Security Callbacks 495
Implications of Multiple RPC Servers in a Single Process 497
Use Mainstream Protocols 499
Secure DCOM Best Practices 499
DCOM Basics 500
Application-Level Security 502
DCOM User Contexts 502
Programmatic Security 505
Sources and Sinks 508
An ActiveX Primer 509
Secure ActiveX Best Practices 509
What ActiveX Components Are Safe for Initialization and Safe for Scripting? 510
Best Practices for Safe for Initialization and Scripting 511
Summary 515

17 Protecting Against Denial of Service Attacks 517
Application Failure Attacks 517
CPU Starvation Attacks 521
Memory Starvation Attacks 529
Resource Starvation Attacks 530
18 Writing Secure .NET Code

Code Access Security: In Pictures
FxCop: A "Must-Have" Tool
Assemblies Should Be Strong-Named
  Strong-Named Assemblies and ASP.NET
Specify Assembly Permission Requirements
  Request Minimal Permission Set
  Refuse Unneeded Permissions
  Request Optional Permissions
Overzealous Use of Assert
Further Information Regarding Demand and Assert
Keep the Assertion Window Small
Demands and Link Demands
  An Example LinkDemand Security Bug
Use SuppressUnmanagedCodeSecurityAttribute with Caution
Remoting Demands
Limit Who Uses Your Code
No Sensitive Data in XML or Configuration Files
Review Assemblies That Allow Partial Trust
Check Managed Wrappers to Unmanaged Code for Correctness
Issues with Delegates
Issues with Serialization
The Role of Isolated Storage
Disable Tracing and Debugging Before Deploying ASP.NET Applications
Do Not Issue Verbose Error Information Remotely
Deserializing Data from Untrusted Sources
Don't Tell the Attacker Too Much When You Fail
Summary

PART IV SPECIAL TOPICS

19 Security Testing
The Role of the Security Tester
Security Testing Is Different
Building Security Test Plans from a Threat Model
  Decompose the Application
  Identify the Component Interfaces
  Rank the Interfaces by Potential Vulnerability
  Ascertain the Data Structures Used by Each Interface
Attacking Applications with STRIDE
Attacking with Data Mutation

BUY ONLINE AT: http://www.itgovernance.co.uk/products/747
Before Testing 587
Building Tools to Find Flaws 588
Testing Clients with Rogue Servers 606
Should a User See or Modify That Data? 607
Testing with Security Templates 607
When You Find a Bug, You’re Not Done! 609
Test Code Should Be of Great Quality 610
Test the End-to-End Solution 611
Determining Attack Surface 611
Determine Root Attack Vectors 611
Determine Bias For Attack Vectors 612
Count the Biased Vectors in the Product 612
Summary 613

20 Performing a Security Code Review 615
Dealing with Large Applications 617
A Multiple-Pass Approach 618
Low-Hanging Fruit 619
Integer Overflows 620
A Related Issue: Integer Underflows 624
Checking Returns 624
Perform an Extra Review of Pointer Code 625
Never Trust the Data 625
Summary 626

21 Secure Software Installation 627
Principle of Least Privilege 628
Clean Up After Yourself! 630
Using the Security Configuration Editor 630
Low-Level Security APIs 638
Using the Windows Installer 638
Summary 640

22 Building Privacy into Your Application 641
Malicious vs. Annoying Invasions of Privacy 642
Major Privacy Legislation 643
Personally Identifiable Information 643
The EU Directives on Data Protection 643
Safe Harbor Principles 644
Other Privacy Legislation 646
Privacy vs. Security 646
Building a Privacy Infrastructure 647
The Role of the Chief Privacy Officer 648
The Role of the Privacy Advocate 648
Designing Privacy-Aware Applications 649
23 General Good Practices

Don't Tell the Attacker Anything
Service Best Practices
   Security, Services, and the Interactive Desktop
   Service Account Guidelines
Don't Leak Information in Banner Strings
Be Careful Changing Error Messages in Fixes
Double-Check Your Error Paths
Keep It Turned Off!
Kernel-Mode Mistakes
   High-Level Security Issues
   Handles
   Symbolic Links
   Quota
   Serialization Primitives
   Buffer-Handling Issues
   IRP Cancellation
Add Security Comments to Code
Leverage the Operating System
Don't Rely on Users Making Good Decisions
Calling CreateProcess Securely
   Do Not Pass NULL for lpApplicationName
   Use Quotes Around the Path to Executable in lpCommandLine
Don't Create Shared/Writable Segments
Using Impersonation Functions Correctly
Don't Write User Files to \Program Files
Don't Write User Data to HKLM
Don't Open Objects for FULL_CONTROL or ALL_ACCESS
Object Creation Mistakes
Care and Feeding of CreateFile
Creating Temporary Files Securely
Implications of Setup Programs and EFS
File System Reparse Point Issues
Client-Side Security Is an Oxymoron
Samples Are Templates
Dogfood Your Stuff!
You Owe It to Your Users If.
Determining Access Based on an Administrator SID
Allow Long Passwords